NASA/CR-2000-210090
ICASE Report No. 2000-14

Model Checking is Refinement -- Relating Büchi Testing and Linear-time Temporal Logic --

Rance Cleaveland
State University of New York at Stony Brook, Stony Brook, New York

Gerald Lüttgen
ICASE, Hampton, Virginia

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA

Operated by Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

Prepared for Langley Research Center
under Contract NAS1-97046

March 2000

MODEL CHECKING IS REFINEMENT*
— RELATING BÜCHI TESTING AND LINEAR–TIME TEMPORAL LOGIC —

RANCE CLEAVELAND† AND GERALD LÜTTGEN‡

Abstract. This paper develops a semantic foundation for reasoning about reactive systems specifications featuring combinations of labeled transition systems and formulas in linear–time temporal logic (LTL). Using Büchi automata as a semantic basis, the paper introduces two refinement preorders based on DeNicola and Hennessy's notion of may– and must–testing. Alternative characterizations for these relations are provided and used to show that the new preorders are conservative extensions of the traditional DeNicola and Hennessy preorders. The paper then establishes a tight connection between LTL formula satisfaction and the Büchi must–preorder. More precisely, it is shown that a labeled transition system satisfies an LTL formula if and only if it refines an appropriately defined Büchi automaton that can be constructed from the formula. Consequently, the Büchi must–preorder allows for a uniform treatment of traditional notions of process refinement and model checking. The implications of the novel theory are illustrated by means of a simple example system, in which some components are specified as transition systems and others as LTL formulas.

Key words. Büchi automata, temporal logic, process algebra, refinement preorder, specification, testing

Subject classification. Computer Science

1. Introduction. Two schools of thought have emerged in the field of formal methods for designing and reasoning about reactive systems. The first advocates the use of *assertional* approaches, in which different formalisms are employed for describing system specifications and implementations. Typically, implementations are given in an operational, programming–oriented notation, while specifications are presented in a declarative, logical style. The semantics of assertions is then applied to determine whether an implementation satisfies its specification. An example for this paradigm is *model checking* [5, 31, 36], where temporal logics are used to specify properties that systems modeled by Kripke structures or labeled transition systems should satisfy. The second school favors *refinement* approaches in which a single formalism that is equipped with a refinement relation is employed to represent a system's specification and implementation. An implementation is deemed correct if it refines its specification. *Process algebras* [19, 27] fall into this classification, with traditional refinement relations being either behavioral equivalences, e.g., bisimulation [27], or preorders, e.g., based on failures or testing [3, 11].

Both paradigms have advantages and disadvantages. Assertional approaches typically allow the formulation of loose specifications which afford implementors great latitude in their design decisions; but they have difficulty in supporting *compositional reasoning*, owing to the fact that the implementation and specification

*This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1–97046 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, Virginia 23681–2199, USA.

†Department of Computer Science, State University of New York at Stony Brook, Stony Brook, New York 11794–4400, USA, e–mail: rance@cs.sunysb.edu. Research support also provided by AFOSR Grant F49620–95–1–0508, ARO Grant P–38682–MA, and NSF Grants CCR–9505562, CCR–9996086 and INT–9996095.

‡ICASE, Mail Stop 132C, NASA Langley Research Center, Hampton, Virginia 23681–2199, USA, e-mail: luettgen@icase.edu.

languages are different. On the other hand, compositionality is a hallmark of refinement approaches, since one may typically refine one part of a system design independently of others. However, refinement–based specifications are often seen as too detailed and, hence, too constraining for implementors. A formalism that marries the benefits of the two styles would have obvious benefits, as the flexibility of assertional specifications could be combined with the virtues of refinement–oriented compositionality. Such a framework would for example permit a project manager to give loose, assertional specifications of different system components to different design teams. If the the composition of the abstract specifications have been determined to satisfy a desired global system specification, the individual, detailed operational component designs returned by the groups would be guaranteed to "compose" correctly.

The goal of this paper is to develop a unified semantic theory for heterogeneous system specifications featuring mixtures of *labeled transition systems* and formulas in *linear–time temporal logic* (LTL). Using Büchi automata [34] and the testing framework of DeNicola and Hennessy [11] as starting points, we approach this problem by developing *Büchi may–* and *must–preorders* that relate Büchi processes on the basis of their responses to *Büchi tests*. For these refinements preorders, we provide alternative characterizations and employ them for proving conservative–extension results regarding DeNicola and Hennessy's testing theory. We then establish the key result of this paper, namely that LTL model checking may be reduced to *refinement checking*. More precisely, a Büchi process B_ϕ can be constructed from an LTL formula ϕ in such a way that a labeled transition system satisfies ϕ if and only if it is larger than B_ϕ for the Büchi must–preorder. Finally, we show that our must–preorder is compositional for a parallel composition operator that is inspired by the one of CCS [27], and illustrate our technical results by a small example featuring the heterogeneous design of a generic communication protocol.

The remainder of this paper is structured as follows. The next section motivates our work by means of an example. Section 3 develops a theory of Büchi testing, including characterizations of the preorders under consideration and their relation to well–established testing preorders. The connection between Büchi must–testing and LTL model checking is investigated in Section 4. The specification framework is then applied to the example in Section 5, while Section 6 discusses related work. Finally, Section 7 contains our conclusions and directions for future work. The proofs of our main theorems are given in the appendix.

2. Motivating Example. As motivation for the work in this paper, consider the design of a very simple communication protocol given in Figure 2.1.

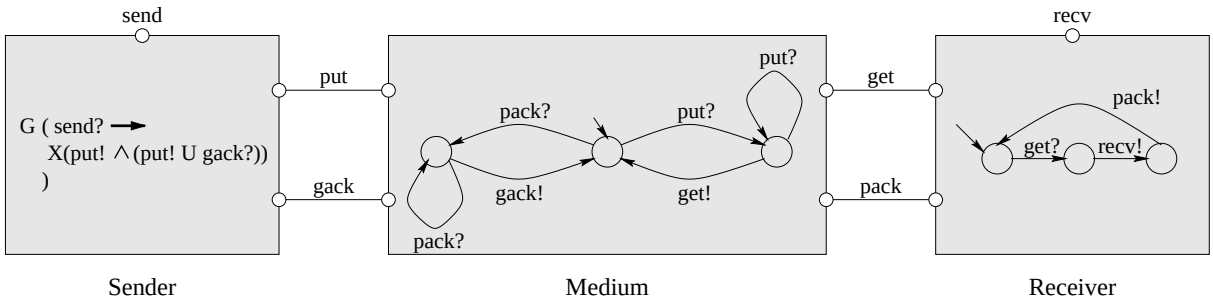

FIG. 2.1. *A simple communication protocol*

The architecture of the protocol has already been fixed by the system designers and consists of a sender `Sender`, a medium `Medium`, and a receiver `Receiver`. The components communicate with the protocol's environment and among themselves via *channels*. In case of component `Sender`, these are the channels

`send`, `put`, and `gack` (*get acknowledgment*). We use the notation `ch?` and `ch!` to indicate the reception and sending of a message from and to channel `ch`, respectively, and refer to these activities as *actions*. Each component in turn has its own specification. `Receiver` and `Medium` are given as labeled transition systems, reflecting the fact that their designs are relatively advanced. The `Sender`, in contrast, is specified assertionally by an LTL formula, i.e., on an abstract specification level. The formula states that whenever a `send?` action occurs during an execution sequence of the sender, the remainder of the execution must begin with a sequence of `put!` actions followed by a `gack?` action.[1] Finally, the overall specification of the protocol's required behavior may be given by the following LTL formula.

$$\texttt{Spec} =_{\text{df}} \texttt{G (send?} \rightarrow \texttt{(F recv!))}$$

This formula encodes a certain reliability guarantee of the protocol regarding the eventual delivery of messages. More precisely, it dictates that in any sequence of actions which the system performs, whenever a `send?` action occurs, a `recv!` action eventually follows. An obvious question that a designer would be interested in is whether the specification of the sender is "strong enough" to ensure that the protocol satisfies `Spec`. The theory developed in this paper provides the semantic framework for answering this question.

A positive answer should be preserved when `Sender` is refined by a labeled transition system satisfying its LTL formula given in Figure 2.1, such as the one depicted on the right. For this to be the case, the underlying refinement relation must be compatible with LTL satisfaction. Moreover, it must be compositional, since `Sender` cannot be considered in isolation, but is just one component of a larger system. Again, the theory to be developed will support such a notion of refinement.

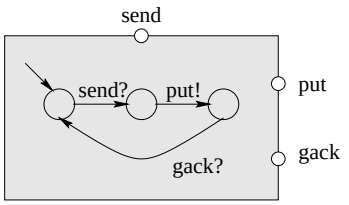

FIG. 2.2. *Refinement of* `Sender`

3. A Theory of Büchi Testing. In this section we extend the *testing theory* of DeNicola and Hennessy [11], which was developed for labeled transition systems in a process–algebraic setting [11], to *Büchi automata*. Traditional testing relates labeled transition systems with respect to their responses to tests via two preorders, the *may–* and *must–preorders*, which distinguish whether systems may or must pass the considered tests. The must–preorder has proved especially interesting because of various full–abstractness results that have been established for it [26] and also because it is compositional with respect to a number of different process constructs, including the parallel operators in Milner's CCS [27] and Hoare's CSP [19].

In this paper, we use Büchi automata as a basis for reasoning about mixed operational and assertional specifications. These automata extend labeled transition systems by means of an acceptance condition for infinite traces. However, the traditional Büchi semantics, which identifies automata having the same infinite languages, is in general not compositional with respect to parallel composition operators, since it is insensitive to the potential for deadlock. Our testing semantics is intended to overcome this problem. In the sequel, we refer to Büchi automata as *Büchi processes* to emphasize that we are equipping Büchi automata with a different semantics than the traditional one. In what follows, we first define Büchi processes and several notions of traces and languages. We then introduce our notion of *Büchi testing*, develop Büchi may– and must–preorders, establish alternative characterizations for the preorders, and show them to be conservative extensions of DeNicola and Hennessy's may– and must–preorders.

[1] In this paper, we assume that LTL formulas are interpreted with respect to sequences of *actions* rather than sequences of states, as is traditionally the case [30]. In formulas, we use actions a as atomic propositions, where a sequence of actions satisfies proposition a if its first element is action a. The adaptation of the LTL semantics is straightforward (cf. Section 4).

3.1. Basic Definitions. Our semantic framework is defined relative to some *alphabet* $\mathcal{A}$, i.e., a countable set of *actions* which does not include the distinguished *unobservable, internal action* τ. In the sequel, we let $a, b, \ldots$ range over $\mathcal{A}$ and $\alpha, \beta, \ldots$ over $\mathcal{A} \cup \{\tau\}$. Büchi processes are distinguished from labeled transition systems in their treatment of infinite traces. Whereas in labeled transition systems all infinite traces are typically deemed possible, in Büchi processes only those infinite traces that go through designated *Büchi states* infinitely often are considered actual executions.

DEFINITION 3.1 (Büchi process & labeled transition system). *A* Büchi process *is a tuple* $\langle P, \longrightarrow, \surd, p\rangle$, *where* P *is a countable set of* states, $\longrightarrow \subseteq P \times (\mathcal{A} \cup \{\tau\}) \times P$ *is the* transition relation, $\surd \subseteq S$ *is the* Büchi set, *and* $p \in P$ *is the* start state. *If* $\surd = P$ *we refer to the Büchi process as* labeled transition system, *in accordance with standard terminology.*

For convenience, we often write (i) $p' \xrightarrow{\alpha} p''$ instead of $\langle p', \alpha, p''\rangle \in \longrightarrow$, (ii) $p' \xrightarrow{\alpha}$ for $\exists p'' \in P.\, p' \xrightarrow{\alpha} p''$, (iii) $p' \longrightarrow$ for $\exists \alpha \in \mathcal{A} \cup \{\tau\},\, p'' \in P.\, p' \xrightarrow{\alpha} p''$, and (iv) $p' \surd$ for $p' \in \surd$. If no confusion arises, we abbreviate the Büchi process $\langle P, \longrightarrow, \surd, p\rangle$ by its start state p and refer to its transition relation and Büchi set as $\longrightarrow_p$ and $\surd_p$, respectively. Moreover, we denote the set of all Büchi processes by $\mathcal{P}$. Note that we do not require Büchi processes to be *finite*–state.

DEFINITION 3.2 (Path & trace). *Let* $\langle P, \longrightarrow, \surd, p\rangle$ *be a Büchi process. A* path π *starting from state* $p' \in P$ *is a potentially infinite sequence* $(\langle p_{i-1}, \alpha_i, p_i\rangle)_{0<i\leq k}$, *where* $k \in \mathbb{N} \cup \{\infty\}$, *such that* $k = 0$, *or* $p_0 = p'$ *and* $p_{i-1} \xrightarrow{\alpha_i} p_i$, *for all* $0 < i \leq k$. *We use* $|\pi|$ *to refer to* k, *the length of* π. *If* $|\pi| = \infty$, *we say that* π *is* infinite; *otherwise,* π *is* finite. *If* $|\pi| \in \mathbb{N}$ *and* $p_{|\pi|} \not\longrightarrow$, *i.e.,* $p_{|\pi|}$ *is a* deadlock *state, path* π *is called* maximal. *Path* π *is referred to as a* Büchi path *if* $|\pi| = \infty$ *and* $|\{i \in \mathbb{N} \,|\, p_i \surd\}| = \infty$. *The (visible)* trace $\mathsf{trace}(\pi)$ *of* π *is defined as the sequence* $(\alpha_i)_{i \in I_\pi} \in \mathcal{A}^* \cup \mathcal{A}^\infty$, *where* $I_\pi =_{df} \{0 < i \leq |\pi| \,|\, \alpha_i \neq \tau\}$.

We denote the sets of all finite paths, all maximal paths, and all Büchi paths starting from state $p' \in P$ by $\Pi_{\mathsf{fin}}(p')$, $\Pi_{\mathsf{max}}(p')$, and $\Pi_{\mathsf{B}}(p')$, respectively. The empty path π with $|\pi| = 0$ is symbolized by $()$ and its empty trace by ϵ. We sometimes write α for trace (α) and use the notation $p' \overset{w}{\Longrightarrow}_p p''$ to indicate that state p' of Büchi process p may evolve to state p'' when observing trace w for some path $\pi \in \Pi_{\mathsf{fin}}(p')$. Formally, $p' \overset{w}{\Longrightarrow}_p p''$ if $\exists \pi = (\langle p_{i-1}, \alpha_i, p_i\rangle)_{0<i\leq k} \in \Pi_{\mathsf{fin}}(p).\, p_0 = p'$, $p_k = p''$, and $\mathsf{trace}(\pi) = w$. We may also introduce different languages for Büchi process p.

$$\begin{array}{lll}
\mathcal{L}_{\mathsf{fin}}(p) & =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{fin}}(p)\} & \subseteq \mathcal{A}^* \qquad \textit{finite–trace language of } p \\
\mathcal{L}_{\mathsf{max}}(p) & =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{max}}(p)\} & \subseteq \mathcal{A}^* \qquad \textit{maximal–trace language of } p \\
\mathcal{L}_{\mathsf{B}}(p) & =_{\mathrm{df}} \{\mathsf{trace}(\pi) \,|\, \pi \in \Pi_{\mathsf{B}}(p)\} & \subseteq \mathcal{A}^* \cup \mathcal{A}^\infty \qquad \textit{Büchi–trace language of } p
\end{array}$$

We also let $\mathcal{I}_p(p') =_{\mathrm{df}} \{a \in \mathcal{A} \,|\, \exists p''.\, p' \overset{a}{\Longrightarrow}_p p''\}$ be the set of initial actions of p in state $p' \in P$.

A key notion for any theory of testing is a system's ability to *diverge*, i.e., to engage in an infinite internal computation [17]. We say that state p' of Büchi process p is *Büchi divergent* or simply *divergent*, in signs $p' \Uparrow_p$, if $\exists \pi \in \Pi_{\mathsf{B}}(p').\, \mathsf{trace}(\pi) = \epsilon$. State p' is called w*–divergent* for some $w = (a_i)_{0<i\leq k} \in \mathcal{A}^* \cup \mathcal{A}^\infty$ if one can reach a divergent state starting from p' when executing a finite prefix of w, i.e., if $\exists l \in \mathbb{N},\, p'' \in P.\, l \leq k$, $p' \overset{w'}{\Longrightarrow} p''$, and $p'' \Uparrow_p$, where $w' =_{\mathrm{df}} (a_i)_{0<i\leq l}$. For convenience, we write $\mathcal{L}_{\mathsf{div}}(p')$ for the *divergent–trace language* of p', i.e., $\mathcal{L}_{\mathsf{div}}(p') =_{\mathrm{df}} \{w \in \mathcal{A}^* \cup \mathcal{A}^\infty \,|\, p' \Uparrow_p w\}$. State p' is convergent or w–convergent, in signs $p' \Downarrow_p$ and $p' \Downarrow_p w$, if not $p' \Uparrow_p$ and not $p' \Uparrow_p w$, respectively. Note that a finite trace $w \in \mathcal{L}_{\mathsf{B}}(p)$ indicates that p is divergent exactly after executing w. In the following, we often omit the indices of the divergence and convergence predicates, as well as of the transition relations, whenever these are obvious from the context. Finally, we write $w \cdot w'$ for the *concatenation* of finite trace $w \in \mathcal{A}^*$ with the finite or infinite trace $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$.

3.2. Testing Theory. The traditional testing framework of DeNicola and Hennessy defines behavioral preorders that relate labeled transition systems with respect to their responses to *tests* [11]. Tests are employed to witness the external interactions a system may have with its environment. In our setting, a test is a Büchi process where certain states are considered to be *success* states. In order to determine whether a system passes a test, one has to examine the finite and infinite *computations* that result when the test runs in lock–step with the system under consideration.

DEFINITION 3.3 (Test, computation, & success).

1. *A Büchi test* $\langle T, \longrightarrow, \surd, t, \mathsf{Suc}\rangle$ *is a Büchi process* $\langle T, \longrightarrow, \surd, t\rangle$ *together with a set* $\mathsf{Suc} \subseteq T$ *of* success states. *If* $\surd = \emptyset$, *we call the test* classical. *The set of all Büchi tests is denoted by* $\mathcal{T}$.
2. *A* potential computation *c with respect to a Büchi process p and a Büchi test t is a potentially infinite sequence* $(\langle p_{i-1}, t_{i-1}\rangle \stackrel{\alpha_i}{\longmapsto}_{r_i} \langle p_i, t_i\rangle)_{0<i\leq k}$, *where* $k \in \mathbb{N} \cup \{\infty\}$, *such that (1)* $p_i \in P$ *and* $t_i \in T$, *for all* $0 \leq i \leq k$, *and (2)* $\alpha_i \in \mathcal{A} \cup \{\tau\}$ *and* $r_i \in \{\blacktriangleleft, \blacktriangleright, \blacklozenge\}$, *for all* $0 < i \leq k$. *The relation* $\longmapsto$ *is defined by the following rules.*

   - $\langle p_{i-1}, t_{i-1}\rangle \stackrel{\alpha_i}{\longmapsto}_{\blacktriangleleft} \langle p_i, t_i\rangle$ *if* $\alpha_i = \tau$, $t_{i-1} = t_i$, $p_{i-1} \stackrel{\tau}{\longrightarrow}_p p_i$, *and* $t_{i-1} \notin \mathsf{Suc}$.
   - $\langle p_{i-1}, t_{i-1}\rangle \stackrel{\alpha_i}{\longmapsto}_{\blacktriangleright} \langle p_i, t_i\rangle$ *if* $\alpha_i = \tau$, $p_{i-1} = p_i$, $t_{i-1} \stackrel{\tau}{\longrightarrow}_t t_i$, *and* $t_{i-1} \notin \mathsf{Suc}$.
   - $\langle p_{i-1}, t_{i-1}\rangle \stackrel{\alpha_i}{\longmapsto}_{\blacklozenge} \langle p_i, t_i\rangle$ *if* $\alpha_i \in \mathcal{A}$, $p_{i-1} \stackrel{\alpha_i}{\longrightarrow}_p p_i$, $t_{i-1} \stackrel{\alpha_i}{\longrightarrow}_t t_i$, *and* $t_{i-1} \notin \mathsf{Suc}$.

   c is finite, *in signs* $|c| < \infty$, *if* $k \in \mathbb{N}$. *Otherwise, it is* infinite, *i.e.,* $|c| = \infty$. *The projection* $\mathsf{proj}_p(c)$ *of c on p is defined as* $(\langle p_{i-1}, \alpha_i, p_i\rangle)_{i\in I_p^c} \in \Pi(p)$, *where* $I_p^c =_{df} \{0 < i \leq k \mid r_i \in \{\blacktriangleleft, \blacklozenge\}\}$, *and the projection* $\mathsf{proj}_t(c)$ *of c on t as* $(\langle t_{i-1}, \alpha_i, t_i\rangle)_{i\in I_t^c} \in \Pi(p)$, *where* $I_t^c =_{df} \{0 < i \leq k \mid r_i \in \{\blacktriangleright, \blacklozenge\}\}$. *A potential computation c is called* computation, *if it satisfies the following properties: (1) c is maximal, i.e., $k \in \mathbb{N}$ implies* $p_k \stackrel{\tau}{\not\longrightarrow}_p$, $t_k \stackrel{\tau}{\not\longrightarrow}_t$, *and* $\mathcal{I}_p(p_k) \cap \mathcal{I}_t(t_k) = \emptyset$, *and (2) $k = \infty$ implies* $\mathsf{proj}_p(c) \in \Pi_B(p)$. *The set of all computations of p and t is denoted by* $\mathcal{C}(p, t)$.
3. *Computation c is called* successful *if* $t_{|c|} \in \mathsf{Suc}$, *in case* $|c| < \infty$, *or if* $\mathsf{proj}_t(c) \in \Pi_B(t)$, *in case* $|c| = \infty$. *We say that p* may pass *t, if there exists a successful computation* $c \in \mathcal{C}(p, t)$. *Analogously, p* must pass *t, if every computation* $c \in \mathcal{C}(p, t)$ *is successful.*

Intuitively, an infinite computation of process p and test t differs from an infinite potential computation in that in the former the process is required to enter a Büchi state infinitely often. An infinite computation is then successful if the test also passes through a Büchi state infinitely often. Hence, in contrast with the original theory of DeNicola and Hennessy, some infinite computations can be successful in our setting. Since Büchi processes and Büchi tests potentially exhibit nondeterministic behavior, one may distinguish between the *possibility* and *inevitability* of success. This is captured in the following definitions of the Büchi *may*– and *must*–preorders.

DEFINITION 3.4 (Büchi Testing Preorders). *Let p and q be Büchi processes. Then we define*

- $p \sqsubseteq_{CL}^{\mathsf{may}} q$ *if* $\forall t \in \mathcal{T}.\ p\,\mathsf{may}_{\mathrm{CL}}\,t$ *implies* $q\,\mathsf{may}_{\mathrm{CL}}\,t$.
- $p \sqsubseteq_{CL}^{\mathsf{must}} q$ *if* $\forall t \in \mathcal{T}.\ p\,\mathsf{must}_{\mathrm{CL}}\,t$ *implies* $q\,\mathsf{must}_{\mathrm{CL}}\,t$.

It is straightforward to check that the relations $\sqsubseteq_{CL}^{\mathsf{may}}$ and $\sqsubseteq_{CL}^{\mathsf{must}}$ on $\mathcal{P}$ are preorders, i.e., that they are reflexive and transitive relations. The classical may– and must–preorders of DeNicola and Hennessy are defined analogously, but on labeled transition systems and when restricting $\mathcal{T}$ to classical tests [11].

3.3. Alternative Characterizations. In the following, we present alternative characterizations of the Büchi may– and must–preorders. The characterizations are similar in style to the ones developed by DeNicola and Hennessy and provide the basis for comparing their testing theory to our Büchi testing.

THEOREM 3.5. *Let p and q be Büchi processes. Then*

1. $p \sqsubseteq_{CL}^{may} q$ *if and only if* $\mathcal{L}_{fin}(p) \subseteq \mathcal{L}_{fin}(q)$ *and* $\mathcal{L}_{B}(p) \subseteq \mathcal{L}_{B}(q)$.
2. $p \sqsubseteq_{CL}^{must} q$ *if and only if for all* $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$ *such that* $p \Downarrow w$, *the following hold:*
   (a) $q \Downarrow w$
   (b) $\underline{|w| < \infty\text{:}}$ $\forall q'.\ q \stackrel{w}{\Longrightarrow} q'$ *implies* $\exists p'.\ p \stackrel{w}{\Longrightarrow} p'$ *and* $\mathcal{I}_p(p') \subseteq \mathcal{I}_q(q')$.
   $\underline{|w| = \infty\text{:}}$ $w \in \mathcal{L}_{B}(q)$ *implies* $w \in \mathcal{L}_{B}(p)$.

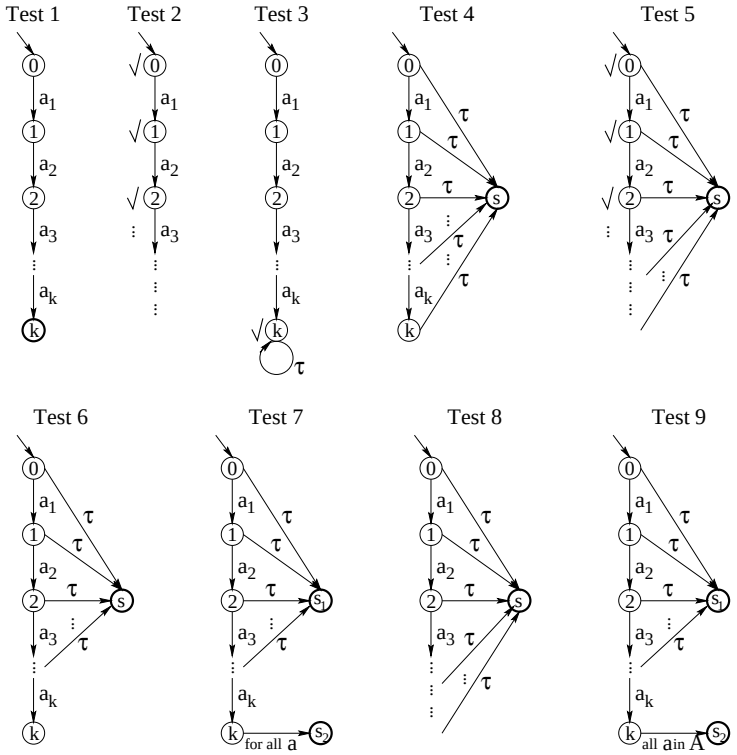

FIG. 3.1. *Büchi tests used for characterizing the Büchi may– and must–preorders*

With respect to finite traces, the characterizations are virtually the same as the ones of DeNicola and Hennessy's preorders [11]. However, we needed to refine the classical characterizations in order to capture the sensitivity of Büchi may– and must–testing to infinite traces. The proof of the above characterization theorem relies on the properties of the following specific Büchi tests.

1. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$, let $t_w^{\mathsf{may},*} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{k\}\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\}$.
2. For $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^\infty$, let $t_w^{\mathsf{may},\infty} =_{\mathrm{df}} \langle T, \longrightarrow, T, 0, \emptyset\rangle$, where $T =_{\mathrm{df}} \mathbb{N}_0$, $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid i \in \mathbb{N}\}$.
3. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$, let $t_w^{\mathsf{may},\mathsf{div}} =_{\mathrm{df}} \langle T, \longrightarrow, \{k\}, 0, \emptyset\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\}$, $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\} \cup \{\langle k, \tau, k\rangle\}$.
4. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$, let $t_w^{\Downarrow} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{s\}\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\} \uplus \{s\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\} \cup \{\langle i, \tau, s\rangle \mid 0 \leq i \leq k\}$.
5. For $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^\infty$, let $t_w^{\Downarrow} =_{\mathrm{df}} \langle T, \longrightarrow, T \setminus \{s\}, 0, \{s\}\rangle$, where $T =_{\mathrm{df}} \mathbb{N}_0 \uplus \{s\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid i \in \mathbb{N}\} \cup \{\langle i, \tau, s\rangle \mid i \in \mathbb{N}_0\}$.
6. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$, let $t_w^{\mathsf{must},*} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{s\}\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\} \uplus \{s\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\} \cup \{\langle i, \tau, s\rangle \mid 0 \leq i < k\}$.

7. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$, let $t_w^{\mathsf{must,max}} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{s_1, s_2\}\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\} \uplus \{s_1, s_2\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\} \cup \{\langle i, \tau, s_1\rangle \mid 0 \leq i < k\} \cup \{\langle k, a, s_2\rangle \mid a \in \mathcal{A}\}$.
8. For $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^\infty$, we define $t_w^{\mathsf{must},\infty} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{s\}\rangle$, where $T =_{\mathrm{df}} \mathbb{N}_0 \uplus \{s\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid i \in \mathbb{N}\} \cup \{\langle i, \tau, s\rangle \mid i \in \mathbb{N}_0\}$.
9. For $w = (a_i)_{0<i\leq k} \in \mathcal{A}^*$ and $A \subseteq \mathcal{A}$, let $t_{w,A}^{\mathsf{must}} =_{\mathrm{df}} \langle T, \longrightarrow, \emptyset, 0, \{s_1, s_2\}\rangle$, where $T =_{\mathrm{df}} \{0, 1, \ldots, k\} \uplus \{s_1, s_2\}$ and $\longrightarrow =_{\mathrm{df}} \{\langle i-1, a_i, i\rangle \mid 0 < i \leq k\} \cup \{\langle i, \tau, s_1\rangle \mid 0 \leq i < k\} \cup \{\langle k, a, s_2\rangle \mid a \in A\}$.

In order to increase comprehension, we also graphically depict the Büchi tests in Figure 3.1. Here, Büchi states are marked by the symbol $\surd$ and success states are distinguished from regular states by thick borders. Intuitively, while Büchi tests $t_w^{\mathsf{may},*}$ and $t_w^{\mathsf{may},\infty}$ test for the presence of finite and Büchi trace w, respectively, Büchi tests $t_w^{\mathsf{may},\mathsf{div}}$ and $t_w^{\Downarrow}$ are capable of detecting divergent behavior when executing trace w. Büchi tests $t_w^{\mathsf{must},*}$, $t_w^{\mathsf{must,max}}$, and $t_w^{\mathsf{must},\infty}$ are concerned with the absence of finite trace, maximal trace, and Büchi trace w, respectively. Finally, Büchi test $t_{w,A}^{\mathsf{must}}$ is capable of comparing the initial action sets of states reached when executing trace w with respect to set $A \subseteq \mathcal{A}$.

Our specific Büchi tests satisfy the following desired properties. Their proofs are simple analyses of the potential computations arising when running the Büchi tests in lock–step with arbitrary Büchi processes.

LEMMA 3.6. *Let p be a Büchi process.*

1. *Let $w \in \mathcal{A}^*$. Then, $w \in \mathcal{L}_{\mathit{fin}}(p)$ if and only if p $\mathsf{may}_{\mathrm{CL}}$ $t_w^{\mathsf{may},*}$.*
2. *Let $w \in \mathcal{A}^\infty$. Then, $w \in \mathcal{L}_{\mathcal{B}}(p)$ if and only if p $\mathsf{may}_{\mathrm{CL}}$ $t_w^{\mathsf{may},\infty}$.*
3. *Let $w \in \mathcal{A}^*$. Then, $w \in \mathcal{L}_{\mathcal{B}}(p)$ if and only if p $\mathsf{may}_{\mathrm{CL}}$ $t_w^{\mathsf{may},\mathsf{div}}$.*
4. *Let $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$. Then, $p \Downarrow w$ if and only if p $\mathsf{must}_{\mathrm{CL}}$ $t_w^{\Downarrow}$.*
5. *Let $w \in \mathcal{A}^*$ such that $p \Downarrow w$. Then, $w \notin \mathcal{L}_{\mathit{fin}}(p)$ if and only if p $\mathsf{must}_{\mathrm{CL}}$ $t_w^{\mathsf{must},*}$.*
6. *Let $w \in \mathcal{A}^*$ such that $p \Downarrow w$. Then, $w \notin \mathcal{L}_{\mathit{max}}(p)$ if and only if p $\mathsf{must}_{\mathrm{CL}}$ $t_w^{\mathsf{must},\mathsf{max}}$.*
7. *Let $w \in \mathcal{A}^\infty$ such that $p \Downarrow w$. Then, $w \notin \mathcal{L}_{\mathcal{B}}(p)$ if and only if p $\mathsf{must}_{\mathrm{CL}}$ $t_w^{\mathsf{must},\infty}$.*

The proof of Theorem 3.5 relies extensively on these intuitive properties of Büchi tests and can be found in Appendix A.1. For finite traces, it proceeds analogously to the corresponding proofs in [11]. For infinite traces, it employs the *infinite–state* tests $t_w^{\mathsf{may},\infty}$, $t_w^{\Downarrow}$, and $t_w^{\mathsf{must},\infty}$ for the "$\Longrightarrow$" proof directions, while the reverse directions can be proved directly along the according definition of successful computation. Note that the usage of infinite–state tests — even when relating finite-state Büchi processes — is justified by our view that Büchi tests represent the arbitrary, potentially irregular behavior of the unknown system environment.

3.4. Conservative Extensions Results. In this section we investigate the relation of our Büchi may– and must–preorders to the corresponding classical preorders, $\sqsubseteq_{DH}^{\mathsf{may}}$ and $\sqsubseteq_{DH}^{\mathsf{must}}$, respectively, as defined by DeNicola and Hennessy [11]. It should be noted that their framework is restricted to *image–finite* labeled transition systems and classical, image–finite tests; a labeled transition system or Büchi process is called image–finite if every state has only a finite number of outgoing transitions for any action.

THEOREM 3.7. *Let p and q be image–finite labeled transition systems.*

1. *If p and q are convergent, then $p \sqsubseteq_{CL}^{\mathit{may}} q$ if and only if $p \sqsubseteq_{DH}^{\mathit{may}} q$.*
2. *$p \sqsubseteq_{CL}^{\mathit{must}} q$ if and only if $p \sqsubseteq_{DH}^{\mathit{must}} q$.*

We refer the reader to Appendix A.2 for the proof of this theorem. In a nutshell, the second part follows by inspection of the alternative characterizations of $\sqsubseteq_{CL}^{\mathsf{must}}$ and $\sqsubseteq_{DH}^{\mathsf{must}}$. The validity of the first part is a consequence of a result established by Narayan Kumar et al. in [28]. They introduced a notion of Büchi

testing for labeled transition systems only, rather than for the more general class of Büchi processes, and they required labeled transition systems and Büchi tests to be convergent and image–finite. Relative to their restricted framework, it is easy to see that our and their definitions of Büchi tests and *passing tests* coincide. Narayan Kumar et al. showed that their Büchi may– and must–preorders coincide with the ones of DeNicola and Hennessy, i.e., (convergent) Büchi tests do not add distinguishing power to classical tests, if only labeled transition systems are taken into account.

Note that Theorem 3.7(1) is invalid if one allows *divergent* labeled transition systems. As a counterexample consider the labeled transition systems $\langle\{p\}, \{\langle p, \tau, p\rangle\}, \{p\}, p\rangle$ and $\langle\{q\}, \emptyset, \{q\}, q\rangle$, as well as the Büchi test $\langle\{t\}, \{\langle t, \tau, t\rangle\}, \{t\}, t, \emptyset\rangle$. Then, $p \sqsubseteq_{DH}^{\mathsf{may}} q$ since $\emptyset = \mathcal{L}_{\mathsf{fin}}(p) \subseteq \mathcal{L}_{\mathsf{fin}}(q) = \{\epsilon\}$, but $p \not\sqsubseteq_{CL}^{\mathsf{may}} q$ since $p \,\mathsf{may}_{\mathrm{CL}}\, t$ and $q \not\mathsf{may}_{\mathrm{CL}}\, t$. The reason for the latter is that the infinite computation $c \in \mathcal{C}(p, t)$, where p and t alternately engage in a τ–transition, is successful. However, the only computation of q and t is the empty computation. This computation is unsuccessful since the set of success states of t is empty.

4. Büchi Must–testing, Trace Inclusion, & Linear–time Temporal Logics. In this section we establish a connection between the Büchi must–preorder $\sqsubseteq_{CL}^{\mathsf{must}}$ and the satisfaction relation $\models$ for linear–time temporal logic (LTL). More specifically, our goal is to show how to construct a Büchi process B_ϕ from an LTL formula ϕ in such a way that $p \models \phi$ if and only if $B_\phi \sqsubseteq_{CL}^{\mathsf{must}} p$, for any labeled transition system p. (Recall that a labeled transition system is a Büchi process in which every state is a Büchi state.) Our result builds on automata–theoretic approaches to LTL model checking developed by Vardi and Wolper [36] and others [6, 15, 20]. These approaches reduce the model–checking problem to one of checking language containment between Büchi automata and rely on the generation of Büchi automata from LTL formulas. To achieve our goal, we first show that $\sqsubseteq_{CL}^{\mathsf{must}}$ coincides with a form of trace inclusion when the lower process is "purely nondeterministic." Then we illustrate how the classical constructions of Büchi automata from LTL formulas may be adapted to cope with the phenomena of deadlock and divergence that labeled transition systems potentially exhibit. In what follows we assume that the set $\mathcal{A}$ of actions is finite.

4.1. Büchi Must–testing & Reverse Trace Inclusion. We start by characterizing the Büchi must–preorder for a certain class of Büchi processes by means of trace inclusion. To state our result, we need to introduce the notion of *pure nondeterminism*. We call a Büchi process p purely nondeterministic, if for all $p' \in P$: (i) $p' \xrightarrow{\tau}_p$ implies $p' \not\xrightarrow{a}_p$, for all $a \in \mathcal{A}$, and (ii) $|\{\langle a, p''\rangle \in \mathcal{A} \times P \mid p' \xrightarrow{a}_p p''\}| = 1$. Note that every Büchi process p can be transformed to a purely nondeterministic Büchi process p', such that $\mathcal{L}_{\mathsf{div}}(p) = \mathcal{L}_{\mathsf{div}}(p')$, $\mathcal{L}_{\mathsf{fin}}(p) = \mathcal{L}_{\mathsf{fin}}(p')$, $\mathcal{L}_{\mathsf{max}}(p) = \mathcal{L}_{\mathsf{max}}(p')$, and $\mathcal{L}_{\mathsf{B}}(p) = \mathcal{L}_{\mathsf{B}}(p')$, by splitting every transition $p' \xrightarrow{a}_p p''$ into two transitions $p' \xrightarrow{\tau}_p p_{\langle p', a, p''\rangle} \xrightarrow{a}_p p''$, where $p_{\langle p', a, p''\rangle} \notin P$ is a new, distinguished state.

THEOREM 4.1. *Let p and q be Büchi processes such that p is purely nondeterministic. Then, $p \sqsubseteq_{CL}^{must} q$ if and only if*

$$\begin{array}{lllll} (i) & \mathcal{L}_{div}(q) & & \subseteq & \mathcal{L}_{div}(p) \\ (ii) & \mathcal{L}_{fin}(q) & \setminus \mathcal{L}_{div}(p) & \subseteq & \mathcal{L}_{fin}(p) \\ (iii) & \mathcal{L}_{max}(q) & \setminus \mathcal{L}_{div}(p) & \subseteq & \mathcal{L}_{max}(p) \\ (iv) & \mathcal{L}_{B}(q) & \setminus \mathcal{L}_{div}(p) & \subseteq & \mathcal{L}_{B}(p) \end{array} \tag{4.1}$$

The proof of the "$\Longrightarrow$"–direction again exploits Lemma 3.6, while the "$\Longleftarrow$"–direction follows by considering the contrapositive. Details can be found in Appendix A.3. The necessity of the premise of this theorem is illustrated by the following example. Consider the Büchi processes $p =_{\mathrm{df}} \langle\{p_1, p_2\}, \{\langle p_1, a, p_1\rangle, \langle p_1, b, p_2\rangle\}, \emptyset, p_1\rangle$ and $q =_{\mathrm{df}} \langle\{q_1, q_2\}, \{\langle q_1, b, q_2\rangle\}, \emptyset, q_1\rangle$. Then p is *not* purely nondeterministic and Equation 4.1 obviously holds, but $p \not\sqsubseteq_{CL}^{\mathsf{must}} q$ since $p \,\mathsf{must}_{\mathrm{CL}}\, t$ and $q \not\mathsf{must}_{\mathrm{CL}}\, t$, for the Büchi test $t =_{\mathrm{df}} \langle\{t_1, t_2\}, \{\langle t_1, a, t_2\rangle\}, \emptyset, t_1, \{t_2\}\rangle$.

4.2. Constructing Büchi Processes from LTL Formulas. We now define the version of LTL that is considered in the sequel and show how an LTL formula may be converted into a purely nondeterministic Büchi process, whose languages contain the traces that satisfy the formula.

4.2.1. Syntax and Semantics of LTL. Our variant of LTL interprets formulas with respect to sequences of actions [14] rather than states [13], since in our setting transitions and not states are labeled. Accordingly, atomic propositions will also be interpreted with respect to actions. Moreover, our variant extends traditional LTL in that its semantics is given with respect to infinite and *finite* traces, i.e., words in $\mathcal{A}^* \cup \mathcal{A}^\infty$ [25]. This permits formulas to constrain ongoing as well as deadlocking behavior. The formal syntax for LTL formulas is defined by the following BNF.

$$\phi \ ::= \ \mathsf{tt} \mid \mathsf{ff} \mid a \mid \neg a \mid \phi \wedge \phi \mid \phi \vee \phi \mid \mathsf{X}\phi \mid \hat{\mathsf{X}}\phi \mid \phi \mathsf{U} \phi \mid \phi \mathsf{V} \phi$$

Here, $a \in \mathcal{A}$ is an atomic proposition that is true of action a and false for all other actions. Moreover, $\hat{\mathsf{X}}$ is the dual of the *next–state* operator X, which in contrast with traditional LTL is not self–dual in our setting since we admit finite traces as models. In the following, we denote the set of all LTL formulas by $\mathcal{F}$. We say that a trace $w = (a_i)_{0<i\leq k} \in \mathcal{A}^* \cup \mathcal{A}^\infty$ *satisfies* ϕ if $w \models \phi$ holds. The relation $\models \subseteq (\mathcal{A}^* \cup \mathcal{A}^\infty) \times \mathcal{F}$ is the least relation satisfying the conditions in Table 4.1, where w_j stands for $(a_i)_{j\leq i\leq k} \in \mathcal{A}^*$, for any $1 \leq j \leq k$. We also say that a Büchi process p satisfies LTL formula ϕ, in signs $p \models \phi$, if $\forall w \in \mathcal{L}_{\mathsf{max}}(p) \cup \mathcal{L}_{\mathsf{B}}(p) \cup \mathcal{L}_{\mathsf{div}}(p).\ w \models \phi$. It should be noted that our syntax limits the application of negation to actions, rather than generally defining a formula $\neg\phi$ with meaning $w \models \neg\phi$ if $w \not\models \phi$. This is not a restriction since our logic is *self–dual*, i.e., the operators $\wedge$ and $\vee$, X and $\hat{\mathsf{X}}$, and U and V are dual to each other.

TABLE 4.1
Semantics of LTL formulas

$w \models \mathsf{tt}$		
$w \models a$	if	$w \neq \epsilon$ and $a_1 = a$
$w \models \neg a$	if	$w \not\models a$
$w \models \phi_1 \wedge \phi_2$	if	$w \models \phi_1$ and $w \models \phi_2$
$w \models \phi_1 \vee \phi_2$	if	$w \models \phi_1$ or $w \models \phi_2$
$w \models \mathsf{X}\phi$	if	$w \neq \epsilon$ and $w_2 \models \phi$
$w \models \hat{\mathsf{X}}\phi$	if	$w \neq \epsilon$ implies $w_2 \models \phi$
$w \models \phi_1 \mathsf{U} \phi_2$	if	$\exists 0 < i \leq k.\ w_i \models \phi_2$ and $\forall 0 < j < i.\ w_j \models \phi_1$
$w \models \phi_1 \mathsf{V} \phi_2$	if	$(\forall 0 < i \leq k.\ w_i \models \phi_2)$ or $(\exists 0 < i \leq k.\ w_i \models \phi_1$ and $\forall 0 < j \leq i.\ w_j \models \phi_2)$

The intuitive meaning of the LTL operators is the following. The symbols tt and ff stand for the propositional constants *true* and *false*, which are satisfied by every trace and no trace, respectively. A finite or infinite trace satisfies the atomic proposition a if the trace is not empty and if its first action is a. It satisfies $\neg a$ if it does not satisfy a. The propositional constructs $\wedge$ and $\vee$ have their usual interpretation as *conjunction* and *disjunction*, respectively. The unary operators X and $\hat{\mathsf{X}}$ represent *next–state* operators. Intuitively, the trace $a \cdot w$ satisfies $\mathsf{X}\phi$ and $\hat{\mathsf{X}}\phi$, if w satisfies ϕ. The only difference between $\mathsf{X}\phi$ and $\hat{\mathsf{X}}\phi$ arises when considering the empty trace ϵ. Whereas ϵ satisfies $\hat{\mathsf{X}}\phi$, it violates $\mathsf{X}\phi$. Formula $\phi_1 \mathsf{U} \phi_2$ represents an *until* property and is satisfied by any trace which satisfies ϕ_1 until ϕ_2 becomes valid. $\phi_1 \mathsf{V} \phi_2$ is a *release* formula and is satisfied by any trace which satisfies ϕ_2 unless this formula is released from its obligation by the truth of ϕ_1, which need never occur. Finally, we may introduce the derived operators G ("generally") and F ("eventually"), already used in Section 2, by defining $\mathsf{G}\phi =_{\mathrm{df}} \mathsf{ff}\,\mathsf{V}\,\phi$ and $\mathsf{F}\phi =_{\mathrm{df}} \mathsf{tt}\,\mathsf{U}\,\phi$.

In the remainder of this section we describe how to construct a purely nondeterministic Büchi process B_ϕ from LTL formula ϕ such that $p \models \phi$ if and only if $B_\phi \sqsubseteq_{CL}^{\mathsf{must}} p$, for any labeled transition system p. We present the construction of B_ϕ in three stages.

4.2.2. Constructing Büchi Processes: Infinite Traces. To begin with, we concentrate on infinite traces and show how to build a convergent Büchi process B_ϕ^1 such that $w \in \mathcal{L}_{\mathsf{B}}(B_\phi^1)$ if and only if $w \in \mathcal{A}^\infty$ and $w \models \phi$. The construction of B_ϕ^1 can be done using existing techniques [9, 15, 36] for converting traditional LTL formulas into Büchi automata. Note that formulas $\mathsf{X}\phi'$ and $\hat{\mathsf{X}}\phi'$ coincide, for any $\phi' \in \mathcal{F}$, when considering only infinite traces as models. Using, e.g., the algorithm of [9], one may build a Büchi automaton whose language contains the infinite traces satisfying ϕ. The states in this automaton are labeled by sets of formulas, and the construction ensures that infinite Büchi traces emanating from a state are guaranteed to satisfy each formula labeling this state. We now may adapt the following classical result.

THEOREM 4.2. *Let ϕ be an LTL formula. Then there exists a Büchi process B_ϕ^1 such that $w \models \phi$ if and only if $w \in \mathcal{L}_B(B_\phi^1)$, for all $w \in \mathcal{A}^\infty$.*

One may immediately derive the following corollary.

COROLLARY 4.3. *Let p be a convergent, deadlock–free labeled transition system, and let ϕ be an LTL formula. Then $p \models \phi$ if and only if $\mathcal{L}_B(p) \subseteq \mathcal{L}_B(B_\phi^1)$.*

4.2.3. Allowing Finite Maximal Traces. In the second stage of our construction of B_ϕ, we show how to generate a Büchi process B_ϕ^2 satisfying $w \models \phi$ if and only if $w \in \mathcal{L}_{\mathsf{B}}(B_\phi^2) \cup \mathcal{L}_{\mathsf{max}}(B_\phi^2)$, for any $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$. The basic approach relies on altering Büchi process B_ϕ^1 to handle finite traces. More precisely, for every state s in B_ϕ^1 we check whether all formulas contained in s are satisfied by the deadlock trace ϵ. Checking for acceptance of the deadlock trace can be done syntactically, along the structure of formulas. Next, for every state s in B_ϕ^1 such that each LTL formula ϕ labeling s is satisfied by ϵ, we add a transition $s \xrightarrow{\tau} \delta$, where δ is a new state that is labeled with $\{\hat{\mathsf{X}}\,\mathsf{ff}\}$, which has ϵ as its only model. However, since we give deadlocks a meaning in form of state δ, we need to eliminate other states having no outgoing transitions in B_ϕ^1. Such states correspond to logical contradictions, i.e., the set of formulas labeling such states is not satisfiable. In B_ϕ^2 we eliminate such deadlock states by removing them from the acceptance set if they are labeled as such, and then adding τ–loops at each of these states.

PROPOSITION 4.4. *Let ϕ be an LTL formula. Then there exists a Büchi process B_ϕ^2 such that:*

1. $\forall w \in \mathcal{A}^*.\ \ w \models \phi$ *if and only if* $w \in \mathcal{L}_{max}(B_\phi^2)$
2. $\forall w \in \mathcal{A}^\infty.\ w \models \phi$ *if and only if* $w \in \mathcal{L}_B(B_\phi^2)$

The second part of the proposition follows immediately from Theorem 4.2, since B_ϕ^2 and B_ϕ^1 possess the same Büchi traces. The first part is a consequence of the fact that (i) our construction ensures that $w \in \mathcal{L}_{\mathsf{max}}(B_\phi^2)$ if and only if $s \xrightarrow{w}_{B_\phi^2} \delta$ and that (ii) $s \xrightarrow{w}_{B_\phi^2} \delta$ holds if and only if $w \models \phi$. As a consequence of this result, we obtain the following theorem.

THEOREM 4.5. *Let p be a convergent labeled transition system, and let ϕ be an LTL formula. Then $p \models \phi$ if and only if $\mathcal{L}_{max}(p) \subseteq \mathcal{L}_{max}(B_\phi^2)$ and $\mathcal{L}_B(p) \subseteq \mathcal{L}_B(B_\phi^2)$.*

4.2.4. Allowing Divergent Traces. As the third step in our construction, we generate a Büchi process B_ϕ^3 that additionally takes divergent traces of labeled transition systems into account. Recall that for general labeled transition systems p we defined $p \models \phi$ if $w \models \phi$ for all $w \in \mathcal{L}_{\mathsf{max}}(p) \cup \mathcal{L}_{\mathsf{B}}(p) \cup \mathcal{L}_{\mathsf{div}}(p)$.

We modify B^2_ϕ to a Büchi process B^3_ϕ by adding divergent states. Intuitively, the divergent states of B^3_ϕ should have the following property. If $w \in \mathcal{A}^*$ is such that $w \cdot w' \models \phi$ for any $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$, then the states reachable in B^3_ϕ via w should be divergent. In essence, divergence is intended to capture tautologies, i.e., LTL formulas satisfied by any trace. The construction of B^3_ϕ relies firstly on the construction of a traditional finite–state machine for recognizing words in $\mathcal{A}^*$ satisfying the aforementioned property. This may be done as follows.

1. Apply the traditional subset construction to determinize B^2_ϕ. The label of each state in the determinized automaton will be a set of sets of LTL formulas.
2. For each state s, check whether the formula $\bigvee_{F \in \ell(s)} \bigwedge_{\phi \in F} \phi$ is a tautology, where $\ell(s)$ is the set of sets of formulas labeling s in the determinized automaton. If so, mark state s as accepting. Note that the tautology check can be performed algorithmically, although a consideration of this point is beyond the scope of this paper.

It can be shown that a finite word $w \in \mathcal{A}^*$ is accepted by the resulting automaton A_ϕ if and only if $w \cdot w' \models \phi$ for any $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$. We may now build B^3_ϕ by first taking the synchronous product of B^2_ϕ and A_ϕ. States in this product have the form $\langle s_B, s_A \rangle$, where s_B is a state in B^2_ϕ and s_A is a state in A_ϕ. Such a state is a Büchi accepting state in B^3_ϕ if s_B is a Büchi state in B^2_ϕ or if s_A is an accepting state in A_ϕ. In the latter case, we make the state divergent by adding a τ–loop to it. We also add a–loops to the state, for every $a \in \mathcal{A}$, as well as a τ–transition to δ. This construction leads to the following lemma and proposition.

LEMMA 4.6. *Let s be the start state of Büchi process B^3_ϕ, and let $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$ be such that $s \Uparrow_{B^{div}_\phi} w$. Then $w \cdot w' \models \phi$, for any $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$.*

PROPOSITION 4.7. *Let $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$. Then $w \models \phi$ if and only if $w \in \mathcal{L}_{max}(B^3_\phi) \cup \mathcal{L}_B(B^3_\phi) \cup \mathcal{L}_{div}(B^3_\phi)$.*

The validity of this proposition is due to Proposition 4.4 when considering that B^3_ϕ possesses by construction the same maximal traces and the same infinite Büchi traces as B^3_ϕ. Thus, only the direction "$\Longleftarrow$" for divergent traces $w \in \mathcal{L}_{\mathsf{div}}(B^3_\phi)$ needs to be established. However, this case is taken care of by Lemma 4.6. Before we can state and prove our main result of this section, we need one more lemma.

LEMMA 4.8. *Let ϕ be an LTL formula, and let p be a labeled transition system such that $p \models \phi$. Then $w \in \mathcal{L}_{div}(p)$ implies $w \in \mathcal{L}_{div}(B^3_\phi)$.*

The proofs of this lemma follows from the fact that if $w \in \mathcal{L}_{\mathsf{div}}(p)$, then there exists a finite prefix w' of w such that $w' \cdot w'' \in \mathcal{L}_{\mathsf{div}}(p)$. This implies that w' must lead to a divergent state in B^3_ϕ. Proposition 4.7 and Lemma 4.8 are the key for proving the following theorem, which lifts Corollary 4.2 and Theorem 4.5 to arbitrary labeled transition systems. Its proof can be found in Appendix A.4.

THEOREM 4.9. *Let p be a labeled transition system and ϕ an LTL formula. Then, $p \models \phi$ if and only if*

$$\begin{array}{lllll}
(i) & \mathcal{L}_{div}(p) & & \subseteq & \mathcal{L}_{div}(B^3_\phi) \\
(ii) & \mathcal{L}_{fin}(p) & \setminus \mathcal{L}_{div}(B^3_\phi) & \subseteq & \mathcal{L}_{fin}(B^3_\phi) \\
(iii) & \mathcal{L}_{max}(p) & \setminus \mathcal{L}_{div}(B^3_\phi) & \subseteq & \mathcal{L}_{max}(B^3_\phi) \\
(iv) & \mathcal{L}_B(p) & \setminus \mathcal{L}_{div}(B^3_\phi) & \subseteq & \mathcal{L}_B(B^3_\phi)
\end{array}$$

Note that the "$\Longrightarrow$" direction of Theorem 4.9 is invalid if p is allowed to be an arbitrary Büchi process. As a counter–example, consider $p =_{\mathrm{df}} \langle \{p_1, p_2, p_3\}, \{\langle p_1, a, p_2\rangle, \langle p_1, b, p_3\rangle, \langle p_3, b, p_3\rangle\}, \emptyset, p_1\rangle$ and $\phi =_{\mathrm{df}} a$. Then $p \models a$, since $b^\infty \notin \mathcal{L}_{\mathsf{B}}(p)$, and $b \in \mathcal{L}_{\mathsf{fin}}(p) \setminus \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. But obviously $b \notin \mathcal{L}_{\mathsf{fin}}(B^3_\phi)$.

4.3. Relating LTL Satisfaction and the Büchi Must–preorder. As the last step in relating the LTL satisfaction relation $\models$ to the Büchi must–preorder $\sqsubseteq_{CL}^{\mathsf{must}}$, we employ B_ϕ^3 to construct a Büchi process B_ϕ such that $p \models \phi$ if and only if $B_\phi \sqsubseteq_{CL}^{\mathsf{must}} p$. We first note that for any ϕ, B_ϕ^3 can be transformed to a purely nondeterministic Büchi process B_ϕ while preserving all languages, as outlined in Section 4.1. Thus, Theorem 4.9 is valid for B_ϕ as well as for B_ϕ^3. By combining Theorems 4.1 and 4.9 we obtain the desired main result as a corollary.

COROLLARY 4.10 (Büchi Must–testing and LTL Model Checking). *Let p be a labeled transition system and ϕ be an LTL formula. Then we have $p \models \phi$ if and only if $B_\phi \sqsubseteq_{CL}^{must} p$.*

As a consequence of this corollary, our notion of Büchi must–testing not only extends DeNicola and Hennessy's must–preorder [11] to Büchi processes, as well as the variant of Büchi must–testing introduced by Kumar et al. [28], but is also compatible with the satisfaction relation of linear–time logics.

5. Motivating Example — Revisited. In this section we illustrate the application of our theory by revisiting and formalizing the motivating example introduced in Section 2. To do so, we need to define two operators on Büchi automata: *parallel composition* and *restriction*.

Our parallel composition operator "|" and the restriction operator $\backslash A$, where $A \subseteq \mathcal{A}$, are inspired by the ones in the process algebra CCS [27]. We assume that alphabet $\mathcal{A}$ is composed of two sets $\mathcal{A}!$ and $\mathcal{A}?$, representing *sending* and *receiving actions*, such that for every $a! \in \mathcal{A}!$ there exists a corresponding $a? \in \mathcal{A}?$, and vice versa. Here, a should be interpreted as a *channel name*. The intuition for parallel composition in CCS is that a process willing to send a message on channel a and another one able to receive a message on a can do so by performing the actions $a!$ and $a?$ in synchrony with each other. This *handshake* is invisible to an external observer, i.e., it results in the distinguished, unobservable action τ. When adapting the CCS parallel operator to our framework of Büchi processes, the questions that naturally arises concerns the interpretation of Büchi traces. We adopt the following point of view: Intuitively, "fair merges" of Büchi processes p and q should also be Büchi traces of $p|q$. Moreover, a Büchi trace of one process, when merged with a finite trace of the other process, should result in a Büchi trace of $p|q$.

Formally, we define the parallel composition of Büchi processes $\langle P, \longrightarrow_p, \surd_p, p\rangle$ and $\langle Q, \longrightarrow_q, \surd_q, q\rangle$ to be the Büchi process $\langle P|Q, \longrightarrow_{p|q}, \surd_{p|q}, p|q\rangle$, where $P|Q =_{\mathrm{df}} \{p'|q' \mid p' \in P, q' \in Q\} \cup \{q'|p' \mid p' \in P, q' \in Q\}$. The transition relation $\longrightarrow_{p|q}$ is the least relation satisfying the following rules.

(1)	$p' \xrightarrow{\alpha}_p p''$	*implies*	$p'\|q' \xrightarrow{\alpha}_{p\|q} q'\|p''$	if	$p'\surd_p$
(2)	$p' \xrightarrow{\alpha}_p p''$	*implies*	$p'\|q' \xrightarrow{\alpha}_{p\|q} p''\|q'$	if	not $p'\surd_p$
(3)	$q' \xrightarrow{\alpha}_q q''$	*implies*	$p'\|q' \xrightarrow{\alpha}_{p\|q} q''\|p'$		
(4)	$p' \xrightarrow{a!}_p p''$ and $q' \xrightarrow{a?}_q q''$	*implies*	$p'\|q' \xrightarrow{\tau}_{p\|q} q''\|p''$	if	$p'\surd_p$
(5)	$p' \xrightarrow{a!}_p p''$ and $q' \xrightarrow{a?}_q q''$	*implies*	$p'\|q' \xrightarrow{\tau}_{p\|q} p''\|q''$	if	not $p'\surd_p$
(6)	$p' \xrightarrow{a?}_p p''$ and $q' \xrightarrow{a!}_q q''$	*implies*	$p'\|q' \xrightarrow{\tau}_{p\|q} q''\|p''$	if	$p'\surd_p$
(7)	$p' \xrightarrow{a?}_p p''$ and $q' \xrightarrow{a!}_q q''$	*implies*	$p'\|q' \xrightarrow{\tau}_{p\|q} p''\|q''$	if	not $p'\surd_p$

These rules are in accordance with our above–mentioned intuition of system behavior. The "switching" of the states of p and q in Rules (1), (3), (4), and (6) allows us to fairly merge "Büchi traces with Büchi traces" and "Büchi traces with finite traces" of the argument Büchi processes. This switching is also done for logical conjunction in the construction of Büchi automata from LTL formulas [9]. Finally, the Büchi predicate $\surd_{p|q}$ is defined by $p'|q'\surd_{p|q}$ if $p'\surd_p$, for any $p' \in P$ and $q' \in Q$. A similar construction could be done for CSP–style

parallel composition [19]. The unary *restriction* operator $\setminus A$, for $A \subseteq \mathcal{A}$, essentially is a *scoping mechanism* on channel names. Intuitively, $p \setminus A$ is defined as the Büchi process p, except that all transitions labeled by actions $a!$ and $a?$, where $a \in A$, are eliminated. One can now obtain the desired compositionality result of the Büchi may– and must–preorders with respect to the new operators.

PROPOSITION 5.1. *Let p_1, p_2, q_1 and q_2 be Büchi processes and $A \subseteq \mathcal{A}$. Then*

(i)	$p_1 \sqsubseteq_{CL}^{may} p_2$	*and*	$q_1 \sqsubseteq_{CL}^{may} q_2$	*implies*	$p_1 \mid q_1 \sqsubseteq_{CL}^{may} p_2 \mid q_2$.
(ii)	$p_1 \sqsubseteq_{CL}^{must} p_2$	*and*	$q_1 \sqsubseteq_{CL}^{must} q_2$	*implies*	$p_1 \mid q_1 \sqsubseteq_{CL}^{must} p_2 \mid q_2$.
(iii)	$p_1 \sqsubseteq_{CL}^{may} p_2$			*implies*	$p_1 \setminus A \sqsubseteq_{CL}^{may} p_2 \setminus A$.
(iv)	$p_1 \sqsubseteq_{CL}^{must} p_2$			*implies*	$p_1 \setminus A \sqsubseteq_{CL}^{must} p_2 \setminus A$.

The proof of this proposition can be done by exploiting the characterizations of the Büchi may– and must–preorders and our conservative extension results, as presented in Sections 3.3 and 3.4. Regarding finite traces, one can then adapt the corresponding proofs of DeNicola and Hennessy [11]. The compositionality with respect to Büchi traces is straightforward regarding the restriction operator; for the parallel operator, it is a consequence of the formalization of our intuition of fair merging.

Let us return to the motivating example of a generic communication protocol. To demonstrate that the LTL specification of the sender is strong enough to ensure that the protocol is correct, in the sense of satisfying the temporal formula $B_{\texttt{Spec}}$ given in Section 2, we may use the results of this paper as follows.

1. Construct the purely nondeterministic Büchi process $B_{\texttt{Spec}}$ for LTL formula `Spec`, as illustrated in Section 4.2.
2. Construct the purely nondeterministic Büchi process $B_{\texttt{Sender}}$ for LTL formula $\phi_{\texttt{sender}}$ describing the behavior of the sender.
3. Assemble the overall system: $\texttt{System} =_{\text{df}} (B_{\texttt{Sender}} \mid \texttt{Medium} \mid \texttt{Receiver}) \setminus \{\texttt{put}, \texttt{get}, \texttt{pack}, \texttt{gack}\}$.
4. Determine whether or not $B_{\texttt{Spec}} \sqsubseteq_{CL}^{must} \texttt{System}$.

In this case, the answer is positive, and Proposition 5.1 guarantees that replacing $B_{\texttt{Sender}}$ with any Büchi process p such that $B_{\texttt{Sender}} \sqsubseteq_{CL}^{must} p$ will ensure that the overall system meets its specification. If p is a labeled transition system then $B_{\texttt{Sender}} \sqsubseteq_{CL}^{must} p$ holds exactly when $p \models \phi_{\texttt{Sender}}$. One example of such a p is the labeled transition system depicted in Figure 2.2.

6. Related Work. Other researchers have also investigated formalisms that permit some form of combined assertional and operational reasoning. Of most direct relevance to this paper is the work of Kurshan [23], who developed a theory of *ω–word automata* that includes notions of synchronous and asynchronous composition. However, his underlying semantic model maps processes to their *maximal (infinite) traces*, and the associated notion of refinement is (reverse) trace inclusion. In theories of concurrency such as CCS [27] and CSP [19], in which deadlock is possible, maximal trace inclusion is not compositional [26]. In contrast, our must–preorder is compositional, at least for the operators presented here. The idea of testing was also adopted by Valmari in [35] where a notion of tester process dealing with finite and infinite traces, divergence, and failures is developed. Other work, such as that of Kupferman and Vardi [22], Grumberg and Long [16], and Clarke, Long and McMillan [7] investigated *modular* and *compositional model–checking* in similar non–deadlock environments. In each case, temporal formulas are used, sometimes in conjunction with additional processes to capture "environmental" information about the module being analyzed. Andersen [1] developed an approach to compositional model checking in which formulas are "factored" by parallel components given as labeled transition systems, yielding new formulas that must be satisfied by the

system comprising the remaining components. His work takes place in the setting of potentially deadlocked processes, although the problem he considered is more narrowly defined than the one studied here.

Relatively more work has been devoted to analyzing relationships between *refinement* and *logical* approaches. One line of study relates temporal–logic specifications to refinement–based ones by establishing that one system refines another if and only if it satisfies the same properties. Results along these lines were pioneered by Hennessy and Milner [18] for *bisimulation* equivalence [27] and a modal logic of their devising [27]. Stirling developed similar characterizations for other refinement orderings and related logics [33]. The ideas were also adopted by Browne, Clarke and Grumberg [4] regarding bisimulation equivalence and the branching–time temporal logic CTL*, by Dams [10] for several variants of the *simulation* preorder [27] and the logic CTL, and by DeNicola and Vaandrager [12] with respect to *branching bisimulation*. Another line of research involves the encoding of labeled transition systems as logical formulas, and vice versa. Steffen and Ingolfsdottir [32] defined an algorithm for converting finite–state labeled transition systems into formulas in the *mu–calculus* [21], while Larsen [24] demonstrated that certain mu–calculus formulas can be encoded as bisimulation–based *implicit specifications*.

Finally, traditional testing has also been enriched with notions of *fairness* [2, 29]. These results, while not addressing the issue of temporal logic, provide an alternative means — besides introducing Büchi states — of incorporating notions of infinite computation into labeled transition systems.

7. Conclusions and Future Work. In this paper we conservatively extended the testing theories of DeNicola and Hennessy [11] and Narayan Kumar et al. [28] to Büchi processes. We illustrated that Büchi processes provide a uniform basis for analyzing heterogeneous reactive–system specifications given as a mixture of labeled transition systems and formulas in linear–time temporal logics (LTL). We then studied the derived Büchi may– and must–preorders, developed alternative characterizations, and showed that the Büchi must–preorder degrades to a variant of reverse trace inclusion when its first argument is purely nondeterministic. Using the latter result, we established that standard algorithms for constructing Büchi processes from LTL formulas can be adapted to our setting in such a way that LTL model checking reduces to checking our form of trace inclusion. In a nutshell, we proved that

$$\textit{LTL model checking} = \textit{Büchi must–preorder checking} + \textit{pure nondeterminism}.$$

Hence, LTL model checking may be viewed as refinement. To illustrate the utility of our novel framework, we presented several operators for constructing specifications, argued that the Büchi must–preorder is substitutive for the operators, and gave an example showing how they may be used to build system designs.

The results of this paper are important first steps towards a more ambitious goal, namely developing languages combining operational and assertional styles of specification. Accordingly, future research should focus on studying languages mixing operators from process algebras and LTL, which can be given a semantics in terms of Büchi processes. For specific languages, one could then study compositionality issues, fully abstractness, and axiomatic characterizations of our Büchi must–preorder, as is usually done in the field of process algebra. For the sake of completing the theory of Büchi testing, we intend to investigate the consequences of restricting our framework to finite–state tests. Moreover, we want to explore how well–known algorithms for computing DeNicola and Hennessy's must–preorder [8] can be lifted to the Büchi must–preorder on finite–state Büchi processes. We would also like to study theories supporting branching–time logics as well.

REFERENCES

[1] H. Andersen, *Partial model checking*, in Tenth Annual Symposium on Logic in Computer Science (LICS '95), San Diego, CA, USA, July 1995, IEEE Computer Society Press, pp. 398–407.

[2] E. Brinksma, A. Rensink, and W. Vogler, *Fair testing*, in Sixth International Conference on Concurrency Theory (CONCUR '95), I. Lee and S. Smolka, eds., Vol. 962 of Lecture Notes in Computer Science, Philadelphia, PA, USA, August 1995, Springer-Verlag, pp. 313–328.

[3] S. Brookes, C. Hoare, and A. Roscoe, *A theory of communicating sequential processes*, Journal of the Association for Computing Machinery, 31 (1984), pp. 560–599.

[4] M. Browne, E. Clarke, and O. Grumberg, *Characterizing finite Kripke structures in propositional temporal logic*, Theoretical Computer Science, 59 (1988), pp. 115–131.

[5] E. Clarke, E. Emerson, and A. Sistla, *Automatic verification of finite-state concurrent systems using temporal logic specifications*, ACM Transactions on Programming Languages and Systems, 8 (1986), pp. 244–263.

[6] E. Clarke, O. Grumberg, and K. Hamaguchi, *Another look at LTL model checking*, in Sixth International Conference on Computer Aided Verification (CAV '94), D. Dill, ed., Vol. 818 of Lecture Notes in Computer Science, Stanford, CA, USA, June 1994, Springer-Verlag, pp. 415–427.

[7] E. Clarke, D. Long, and K. McMillan, *Compositional model checking*, in Fourth Annual Symposium on Logic in Computer Science (LICS '89), Asilomar, CA, USA, June 1989, IEEE Computer Society Press, pp. 353–362.

[8] R. Cleaveland and M. Hennessy, *Testing equivalence as a bisimulation equivalence*, Formal Aspects of Computing, 5 (1993), pp. 1–20.

[9] R. Cleaveland and Y. Xie, *An operational semantics for temporal logic*, tech. report, State University of New York at Stony Brook, NY, USA, January 2000.

[10] D. Dams, *Abstract Interpretation and Partition Refinement for Model Checking*, Ph.D. thesis, Eindhoven University of Technology, The Netherlands, July 1996.

[11] R. DeNicola and M. Hennessy, *Testing equivalences for processes*, Theoretical Computer Science, 34 (1983), pp. 83–133.

[12] R. DeNicola and F. Vaandrager, *Three logics for branching bisimulation*, Journal of the Association for Computing Machinery, 42 (1995), pp. 458–487.

[13] E. Emerson, *Temporal and modal logic*, in Handbook of Theoretical Computer Science, J. van Leeuwen, ed., Vol. B, North-Holland, 1990, pp. 995–1072.

[14] A. Fantechi, S. Gnesi, and G. Ristori, *Model checking for action based logic*, Formal Methods in System Design, 4 (1994), pp. 187–203.

[15] R. Gerth, M. Vardi, and P. Wolper, *Simple on-the-fly automatic verification of linear temporal logic*, in Proceedings of the IFIP Symposium on Protocol Specification, Testing and Verification (PSTV '95), Warsaw, Poland, June 1995, Chapman & Hall, pp. 3–18.

[16] O. Grumberg and D. Long, *Model checking and modular verification*, ACM Transactions on Programming Languages and Systems, 16 (1994), pp. 843–871.

[17] M. Hennessy, *Algebraic Theory of Processes*, MIT Press, Boston, 1988.

[18] M. Hennessy and R. Milner, *Algebraic laws for nondeterminism and concurrency*, Journal of the Association for Computing Machinery, 32 (1985), pp. 137–161.

[19] C. Hoare, *Communicating Sequential Processes*, Prentice Hall, London, UK, 1985.

[20] G. Holzmann, *The model checker Spin*, IEEE Transactions on Software Engineering, 23 (1997), pp. 279–295. Special issue on Formal Methods in Software Practice.

[21] D. Kozen, *Results on the propositional μ-calculus*, Theoretical Computer Science, 27 (1983), pp. 333–354.

[22] O. Kupferman and M. Vardi, *Modular model checking*, in Compositionality: The Significant Difference, W.-P. de Roever, H. Langmaack, and A. Pnueli, eds., Vol. 1536 of Lecture Notes in Computer Science, Bad Malente, Germany, June 1997, Springer-Verlag.

[23] R. Kurshan, *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*, Princeton Series in Computer Science, Princeton University Press, Princeton, NJ, USA, 1994.

[24] K. Larsen, *The expressive power of implicit specifications*, Theoretical Computer Science, 114 (1993), pp. 119–147.

[25] O. Lichtenstein, A. Pnueli, and L. Zuck, *The glory of the past*, in Third Workshop on Logics of Programs, R. Parikh, ed., Vol. 193 of Lecture Notes in Computer Science, Brooklyn, NY, USA, June 1985, Springer-Verlag, pp. 196–218.

[26] M. Main, *Trace, failure and testing equivalences for communicating processes*, International Journal of Parallel Computing, 16 (1987), pp. 383–400.

[27] R. Milner, *Communication and Concurrency*, Prentice Hall, London, UK, 1989.

[28] K. Narayan Kumar, R. Cleaveland, and S. Smolka, *Infinite probabilistic and nonprobabilistic testing*, in 18th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS '98), V. Arvind and R. Ramanujam, eds., Vol. 1530 of Lecture Notes in Computer Science, Chennai, India, December 1998, Springer-Verlag, pp. 209–220.

[29] V. Natarajan and R. Cleaveland, *Divergence and fair testing*, in 22nd International Colloquium on Automata, Languages and Programming (ICALP '95), Z. Fülöp and F. Gécseg, eds., Vol. 944 of Lecture Notes in Computer Science, Szeged, Hungary, July 1995, Springer-Verlag, pp. 684–695.

[30] A. Pnueli, *The temporal logic of programs*, in 18th IEEE Symposium on the Foundations of Computer Science (FOCS '77), Providence, RI, USA, 1977, IEEE Computer Society Press, pp. 46–57.

[31] J. Queille and J. Sifakis, *Specification and verification of concurrent systems in CESAR*, in Fifth International Symposium in Programming, M. Dezani-Ciancaglini and U. Montanari, eds., Vol. 137 of Lecture Notes in Computer Science, Turino, Italy, April 1982, Springer-Verlag, pp. 337–351.

[32] B. Steffen and A. Ingólfsdóttir, *Characteristic formulae for CCS with divergence*, Information and Computation, 110 (1994), pp. 149–163.

[33] C. Stirling, *Modal logics for communicating systems*, Theoretical Computer Science, 49 (1987), pp. 311–347.

[34] W. Thomas, *Automata on infinite objects*, in Handbook of Theoretical Computer Science, J. van Leeuwen, ed., Vol. B, North-Holland, 1990, pp. 133–191.

[35] A. Valmari, *On-the-fly verification with stubborn sets*, in Fifth International Conference on Computer Aided Verification (CAV '93), C. Courcoubetis, ed., Vol. 697 of Lecture Notes in Computer Science, Elounda, Greece, June/July 1993, Springer-Verlag, pp. 397–408.

[36] M. Vardi and P. Wolper, *An automata-theoretic approach to automatic program verification*, in First Annual IEEE Symposium on Logic in Computer Science (LICS '86), Cambridge, MA, USA, June 1986, IEEE Computer Society Press, pp. 332–344.

Appendix A. Proof of the Main Theorems.

A.1. Proof of Theorem 3.5. Let p and q be Büchi processes.

1. For proving the "$\Longrightarrow$"–direction, we distinguish the following cases.
   - $w \in \mathcal{L}_{\mathsf{fin}}(p)$**:** Then $p \,\mathsf{may}_{\mathrm{CL}}\, t_w^{\mathsf{may},*}$ by Lemma 3.6(1) and, since $p \sqsubseteq_{CL}^{\mathsf{may}} q$, also $q \,\mathsf{may}_{\mathrm{CL}}\, t_w^{\mathsf{may},*}$. Applying Lemma 3.6(1) again, we obtain $w \in \mathcal{L}_{\mathsf{fin}}(q)$, as desired.
   - $w \in \mathcal{L}_{\mathsf{B}}(p)$**:** Here, we distinguish the cases $|w| = \infty$ and $|w| < \infty$. In both cases, we closely follow the lines of the first proof part, but use Lemma 3.6(2) and Büchi test $t_w^{\mathsf{may},\infty}$, as well as Lemma (3) and Büchi test $t_w^{\mathsf{may},\mathsf{div}}$, respectively, instead of Lemma 3.6(1) and Büchi test $t_w^{\mathsf{may},*}$.

   For the "$\Longleftarrow$"–direction, assume that t is a Büchi test satisfying $p \,\mathsf{may}_{\mathrm{CL}}\, t$. Then there exists a successful computation $c \in \mathcal{C}(p,t)$ with $w =_{\mathrm{df}} \mathsf{trace}(\mathsf{proj}_p(c)) = \mathsf{trace}(\mathsf{proj}_t(c))$. If $|w| = \infty$ we have $w \in \mathcal{L}_{\mathsf{B}}(p)$. Hence, $w \in \mathcal{L}_{\mathsf{B}}(q)$, and we can construct a successful computation $c' \in \mathcal{C}(q,t)$. The case $|w| < \infty$ is splitted into two sub–cases according to whether $w \in \mathcal{L}_{\mathsf{fin}}(p)$ or $w \in \mathcal{L}_{\mathsf{B}}(p)$. In either case one can easily establish $q \,\mathsf{may}_{\mathrm{CL}}\, t$. Therefore, $p \sqsubseteq_{CL}^{\mathsf{may}} q$, as desired.
2. For the "$\Longrightarrow$"–direction, assume $p \sqsubseteq_{CL}^{\mathsf{must}} q$, and let $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$ such that $p \Downarrow w$.
   (a) Then $p \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\Downarrow}$ by Lemma 3.6(4) and, since $p \sqsubseteq_{CL}^{\mathsf{must}} q$, also $q \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\Downarrow}$. Thus, we obtain $q \Downarrow w$ by applying Lemma 3.6(4) again.
   (b) $|w| < \infty$**:** Let $q \stackrel{w}{\Longrightarrow} q'$ for some q', i.e., $w \in \mathcal{L}_{\mathsf{fin}}(q)$. Assume further that $\nexists p'.\, p \stackrel{w}{\Longrightarrow} p'$ and $\mathcal{I}_p(p') \subseteq \mathcal{I}_q(q')$. We may distinguish the following cases.
      - "$p \stackrel{w}{\not\Longrightarrow}$"**:** Then $w \notin \mathcal{L}_{\mathsf{fin}}(p)$, and by Lemma 3.6(5) we obtain $p \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},*}$. However, $\neg(q \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},*})$ by the same lemma.
      - "$p \stackrel{w}{\Longrightarrow}$"**:** Let $A =_{\mathrm{df}} \{\mathcal{I}_p(p') \mid p \stackrel{w}{\Longrightarrow} p'\} \neq \emptyset$. By assumption, for every $A_i \in A$ there exists an action $a_i \in A_i \setminus \mathcal{I}_q(q')$. Let $B \neq \emptyset$ be the set of these actions. It is easy to see that $p \,\mathsf{must}_{\mathrm{CL}}\, t_{w,B}^{\mathsf{must}}$ due to the construction of Büchi test $t_{w,B}^{\mathsf{must}}$. However, $\neg(q \,\mathsf{must}_{\mathrm{CL}}\, t_{w,B}^{\mathsf{must}})$ since $q' \stackrel{a_i}{\not\Longrightarrow}_q$ for all actions $a_i \in B$.

      Hence, $p \not\sqsubseteq_{CL}^{\mathsf{must}} q$ which is a contradiction.
      $|w| = \infty$**:** Assume $w \notin \mathcal{L}_{\mathsf{B}}(p)$. Then $p \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},\infty}$ by Lemma 3.6(7) and, since $p \sqsubseteq_{CL}^{\mathsf{must}} q$, also $q \,\mathsf{must}_{\mathrm{CL}}\, t_w^{\mathsf{must},\infty}$. But then $w \notin \mathcal{L}_{\mathsf{B}}(q)$ holds by Lemma 3.6(7), as desired.

   For the proof of the "$\Longleftarrow$"–direction, let $t \in \mathcal{T}$ such that $\neg(q \,\mathsf{must}_{\mathrm{CL}}\, t)$, i.e., there exists an unsuccessful computation $c = (\langle\langle q_{i-1}, t_{i-1}\rangle, \alpha_i, \langle q_i, t_i\rangle\rangle)_{0<i\leq k} \in \mathcal{C}(q,t)$. Let $w =_{\mathrm{df}} \mathsf{trace}(\mathsf{proj}_q(c)) = \mathsf{trace}(\mathsf{proj}_t(c))$. If $p \Uparrow w$, we can construct an unsuccessful, infinite computation c' which resembles c until p can engage in its divergent Büchi computation, in which case we can force t not to contribute to c' any more. Thus, c' is an unsuccessful computation, since $\mathsf{proj}_p(c') \in \Pi_{\mathsf{B}}(p)$, but $|\mathsf{proj}_t(c')| < \infty$, i.e., $\mathsf{proj}_t(c') \notin \Pi_{\mathsf{B}}(t)$.

   For the remainder of this proof, let us assume $p \Downarrow w$, i.e., $w \notin \mathcal{L}_{\mathsf{div}}(p)$. According to the definition of (un)successful computations, we distinguish the following two cases.
   - $|c| < \infty$**:** Then $w \in \mathcal{L}_{\mathsf{fin}}(q)$, $q \stackrel{w}{\Longrightarrow} q'$ for some q', and $t_k \notin \mathsf{Suc}$. Due to the maximality of computations we also have $q_k \stackrel{\tau}{\not\longrightarrow}_q$, $t_k \stackrel{\tau}{\not\longrightarrow}_t$, and $\mathcal{I}_q(q_k) \cap \mathcal{I}_t(t_k) = \emptyset$. By Condition 2(a) of the premise (cf., right–hand side of the characterization in Theorem 3.5) we know of the existence of some p' such that $p \stackrel{w}{\Longrightarrow} p'$ and $\mathcal{I}_p(p') \subseteq \mathcal{I}_q(q')$. Using these facts one may construct a finite computation $c' = (\langle\langle p_{i-1}, t'_{i-1}\rangle, \alpha_i, \langle p_i, t'_i\rangle\rangle)_{0<i\leq l} \in \mathcal{C}(p,t)$ with $\mathsf{proj}_t(c') = \mathsf{proj}_t(c)$ and $\langle p_l, t'_l\rangle = \langle p', t_k\rangle$, where $p' \stackrel{\epsilon}{\Longrightarrow}_p p''$ for some $p'' \stackrel{\tau}{\not\longrightarrow}_p$. Note that such a p'' must exist since $p \Downarrow w$. Moreover, $\mathcal{I}_p(p'') \subseteq \mathcal{I}_p(p')$ by the definition of $\mathcal{I}_p(\cdot)$. Because $\mathcal{I}_p(p'') \cap \mathcal{I}_t(t_k) \subseteq \mathcal{I}_q(q') \cap \mathcal{I}_t(t'_l) = \emptyset$

holds, c' cannot be extended. Finally, c' is unsuccessful since $t'_l = t_k \notin \mathsf{Suc}$.

- $|c| = \infty$**:** Hence, $w \in \mathcal{L}_{\mathsf{B}}(q)$ and, since $\mathcal{L}_{\mathsf{B}}(q) \subseteq \mathcal{L}_{\mathsf{B}}(p)$, also $w \in \mathcal{L}_{\mathsf{B}}(p)$. Then it is straightforward to construct an unsuccessful computation $c' \in \mathcal{C}(p,t)$ with $\mathsf{proj}_t(c') = \mathsf{proj}_t(c)$.

In both cases we obtain $\neg(p\,\mathsf{must}_{\mathrm{CL}}\,t)$. Summarizing, we have shown for an arbitrary test $t \in \mathcal{T}$ that $\neg(q\,\mathsf{must}_{\mathrm{CL}}\,t)$ implies $\neg(p\,\mathsf{must}_{\mathrm{CL}}\,t)$, i.e., $p \sqsubseteq_{CL}^{\mathsf{must}} q$, as desired.

This finishes the proof of Theorem 3.5.

A.2. Proof of Theorem 3.7. Consider image–finite labeled transition systems only.

1. Under the additional assumption of *convergence*, the definitions of $\sqsubseteq_{CL}^{\mathsf{may}}$ and the Büchi–may preorder introduced by Narayan Kumar et al. are identical. Narayan Kumar et al. showed their preorder to coincide with $\sqsubseteq_{DH}^{\mathsf{may}}$; hence, also $\sqsubseteq_{CL}^{\mathsf{may}}$ and $\sqsubseteq_{DH}^{\mathsf{may}}$ coincide.

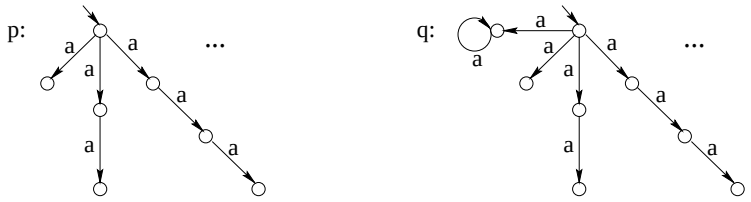

FIG. A.1. *(Counter–)example demonstrating the necessity of the image–finiteness assumption*

2. We now establish $\sqsubseteq_{CL}^{\mathsf{must}} = \sqsubseteq_{DH}^{\mathsf{must}}$ by showing that the alternative characterizations of these preorders coincide when considering the setting of DeNicola and Hennessy. The alternative characterization of $\sqsubseteq_{DH}^{\mathsf{must}}$ (cf. Theorem 3.5(2)) differs from the one of $\sqsubseteq_{CL}^{\mathsf{must}}$ in two ways: (i) the definition of $p \Downarrow w$ and $q \Downarrow w$ also permits the case $w \in \mathcal{A}^{\infty}$, and (ii) Condition (b) in Theorem 3.5(2) for $|w| = \infty$ is missing. Regarding the first point of departure, our definition of divergence implies for all $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^{\infty}$ the following.

$$\forall k \in \mathbb{N}.\ (p \Downarrow w_k \ \text{ implies } \ q \Downarrow w_k) \ \text{ implies } \ (p \Downarrow w \ \text{ implies } \ q \Downarrow w)$$

where $w_k =_{\mathrm{df}} (a_i)_{0<i\leq k} \in \mathcal{A}^*$. Thus, Condition (a) of Theorem 3.5(2) for infinite w is already implied by the same condition for all finite prefixes of w. Moreover, our definition of divergence coincides with the one of DeNicola and Hennessy for labeled transition systems. The second point of departure can be addressed in a similar fashion. In fact, it is easy to establish that the following holds for *image–finite* labeled transition systems p and q and for all $w = (a_i)_{i\in\mathbb{N}} \in \mathcal{A}^{\infty}$ such that $p \Downarrow w$ and $q \Downarrow w$.

$$\forall k \in \mathbb{N}.\ (w_k \in \mathcal{L}_{\mathsf{fin}}(q) \ \text{ implies } \ w_k \in \mathcal{L}_{\mathsf{fin}}(p)) \ \text{ implies } \ (w \in \mathcal{L}_{\mathsf{B}}(q) \ \text{ implies } \ w \in \mathcal{L}_{\mathsf{B}}(p))$$

where $w_k =_{\mathrm{df}} (a_i)_{0<i\leq k} \in \mathcal{A}^*$. Note that in the case where $w \in \mathcal{A}^*$, the w–convergence of q implies $w \notin \mathcal{L}_{\mathsf{B}}(q)$. As a consequence, Condition (a) implies Condition (b) under the assumptions of Theorem 3.7. A (counter–)example demonstrating the necessity of the image–finiteness assumption is depicted in Figure A.1.

Thus, the Büchi may– and must–preorders coincide with DeNicola and Hennessy's may– and must–preorders in the considered setting, as desired.

A.3. Proof of Theorem 4.1. For the proof of the "$\Longrightarrow$"–direction, assume that $p \sqsubseteq_{CL}^{\mathsf{must}} q$, and let $w \in \mathcal{A}^* \cup \mathcal{A}^\infty$. Then

- $w \in \mathcal{L}_{\mathsf{div}}(q)$ if $q \Uparrow w$. By Lemma 3.6(4) we have $\neg(q\ \mathsf{must}_{\mathrm{CL}}\ t_w^{\Downarrow})$. Since $p \sqsubseteq_{CL}^{\mathsf{must}} q$ also $\neg(p\ \mathsf{must}_{\mathrm{CL}}\ t_w^{\Downarrow})$ holds, i.e., $p \Uparrow w$ by applying Lemma 3.6(4) again. Thus, $w \in \mathcal{L}_{\mathsf{div}}(p)$.
- $w \in \mathcal{L}_{\mathsf{fin}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p)$ implies $w \in \mathcal{L}_{\mathsf{fin}}(p)$, $p \Downarrow w$, and also $q \Downarrow w$ by Equation 4.1(i). By Lemma 3.6(5) we conclude $\neg(q\ \mathsf{must}_{\mathrm{CL}}\ t_w^{\mathsf{must},*})$. Because of the premise $p \sqsubseteq_{CL}^{\mathsf{must}} q$ also $\neg(p\ \mathsf{must}_{\mathrm{CL}}\ t_w^{\mathsf{must},*})$ holds, i.e., $w \in \mathcal{L}_{\mathsf{fin}}(p)$ by Lemma 3.6(5).
- The cases $w \in \mathcal{L}_{\mathsf{max}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p)$ and $w \in \mathcal{L}_{\mathsf{B}}(q) \setminus \mathcal{L}_{\mathsf{div}}(p)$ are similar to the previous one but refer to Lemma 3.6(6) and Lemma 3.6(7), respectively. As desired, we may obtain $w \in \mathcal{L}_{\mathsf{max}}(p)$ and $w \in \mathcal{L}_{\mathsf{B}}(p)$, respectively.

Note that this proof direction does not require p to be purely nondeterministic.

For establishing the "$\Longleftarrow$"–direction, assume that the language inclusions of Equation 4.1 hold. Moreover, assume the existence of a Büchi test t such that $\neg(q\ \mathsf{must}_{\mathrm{CL}}\ t)$. Thus, there exists an unsuccessful computation $c = (\langle\langle q_{i-1}, t_{i-1}\rangle, \alpha_i, \langle q_i, t_i\rangle\rangle)_{0<i\leq k} \in \mathcal{C}(q,t)$ with $w =_{\mathrm{df}} \mathsf{trace}(\mathsf{proj}_q(c)) = \mathsf{trace}(\mathsf{proj}_t(c))$. If $p \Uparrow w$, then we can construct an unsuccessful, infinite computation c' which resembles c until p can engage in its divergent Büchi computation, at which point t can be forced to stop contributing to c'. As desired, computation c' is unsuccessful since $\mathsf{proj}_p(c') \in \Pi_{\mathsf{B}}(p)$, but $|\mathsf{proj}_t(c')| < \infty$, i.e., $\mathsf{proj}_t(c') \notin \Pi_{\mathsf{B}}(t)$.

For the remainder of this proof, let us assume $p \Downarrow w$, i.e., $w \notin \mathcal{L}_{\mathsf{div}}(p)$. According to the definition of (un)successful computations, we distinguish the following two cases.

1. $\underline{|c| < \infty}$**:** Here, we have $t_k \notin \mathsf{Suc}$.
   (a) $\underline{w \in \mathcal{L}_{\mathsf{max}}(q)}$**:** By Premise 4.1(iii) we have $w \in \mathcal{L}_{\mathsf{max}}(p)$. Then we can construct a finite computation $c' = (\langle\langle p_{i-1}, t'_{i-1}\rangle, \alpha'_i, \langle p_i, t'_i\rangle\rangle)_{0<i\leq l} \in \mathcal{C}(p,t)$ with $\mathsf{proj}_t(c') = \mathsf{proj}_t(c)$ and $t'_l = t_k$. Thus, c' is unsuccessful, since $|c'| < \infty$ and $t'_l \notin \mathsf{Suc}$.
   (b) $\underline{w \in \mathcal{L}_{\mathsf{fin}}(q) \setminus \mathcal{L}_{\mathsf{max}}(q)}$**:** In this case, we know of the existence of some $a \in \mathcal{A}$ such that $q_k \xrightarrow{a}_q$ and, because of the maximality of computations, $t_k \not\xrightarrow{a}_t$. Thus, $w \cdot a \in \mathcal{L}_{\mathsf{fin}}(q)$ holds, and by Premise 4.1(iv) we have $w \cdot a \in \mathcal{L}_{\mathsf{fin}}(p)$. *Since p is purely nondeterministic*, we may construct a finite computation $c' = (\langle\langle p_{i-1}, t'_{i-1}\rangle, \alpha'_i, \langle p_i, t'_i\rangle\rangle)_{0<i\leq l} \in \mathcal{C}(p,t)$, where $\mathsf{proj}_t(c') = \mathsf{proj}_t(c)$, $t'_l = t_k$ and $p_l \xrightarrow{a}_p$. Indeed, c' is maximal since $t'_l \not\xrightarrow{a}_t$ and $p'_l \not\xrightarrow{b}_p$ for all $b \neq a$. Moreover, c' is unsuccessful, because $|c'| < \infty$ and $t'_l \notin \mathsf{Suc}$.
2. $\underline{|c| = \infty}$**:** Here, $\mathsf{proj}_t(c) \notin \Pi_{\mathsf{B}}(t)$. By Premise 4.1(iv) and since $\mathsf{proj}_q(c) \in \Pi_{\mathsf{B}}(q)$ due to the definition of computation, we have $w \in \mathcal{L}_{\mathsf{B}}(p)$. Hence, we can construct an infinite computation $c' \in \mathcal{C}(p,t)$ such that $\mathsf{proj}_t(c') = \mathsf{proj}_t(c)$. As a consequence, also c' is unsuccessful.

Thus, $\neg(p\ \mathsf{must}_{\mathrm{CL}}\ t)$ and, further, $p \sqsubseteq_{CL}^{\mathsf{must}} q$, as desired.

A.4. Proof of Theorem 4.9. For establishing the "$\Longrightarrow$" direction, let $p \models \phi$, i.e., $w \models \phi$ for all $w \in \mathcal{L}_{\mathsf{max}}(p) \cup \mathcal{L}_{\mathsf{B}}(p) \cup \mathcal{L}_{\mathsf{div}}(p)$. By Proposition 4.7 we also have $w \in \mathcal{L}_{\mathsf{max}}(B^3_\phi) \cup \mathcal{L}_{\mathsf{B}}(B^3_\phi) \cup \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. We may distinguish the following cases.

1. *Case $w \in \mathcal{L}_{div}(p)$:* This case is taken care of by Lemma 4.8.
2. *Case $w \in \mathcal{L}_{fin}(p) \setminus \mathcal{L}_{div}(B^3_\phi)$:* Since p is a labeled transition system, $w \in \mathcal{L}_{\mathsf{fin}}(p)$ is always a finite prefix of a maximal trace or an infinite (Büchi) trace. Hence, we may conclude the existence of some $w' \in \mathcal{A}^* \cup \mathcal{A}^\infty$ such that $w \cdot w' \in \mathcal{L}_{\mathsf{max}}(B^3_\phi) \cup \mathcal{L}_{\mathsf{B}}(B^3_\phi) \cup \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. The other three inclusions, together

with the fact that by construction, every divergent state s in B^3_ϕ satisfies $\mathcal{L}_{\mathsf{max}}(s) = \mathcal{A}^*$, we obtain $w \in \mathcal{L}_{\mathsf{fin}}(B^3_\phi)$, as desired.

3. *Case* $w \in \mathcal{L}_{max}(p) \setminus \mathcal{L}_{div}(B^3_\phi)$*:* Hence, $w \in \mathcal{A}^*$ and, together with Proposition 4.7, $w \in \mathcal{L}_{\mathsf{max}}(B^3_\phi)$.
4. *Case* $w \in \mathcal{L}_B(p) \setminus \mathcal{L}_{div}(B^3_\phi)$*:* Then, $w \in \mathcal{A}^\infty$, and as a consequence of Proposition 4.7, $w \in \mathcal{L}_{\mathsf{B}}(B^3_\phi)$.

Thus the language inclusions stated in equations (i) through (iv) are valid.

For proving the "$\Longleftarrow$" direction, assume that $p \not\models \phi$, i.e., $\exists w \in \mathcal{L}_{\mathsf{max}}(p) \cup \mathcal{L}_{\mathsf{B}}(p) \cup \mathcal{L}_{\mathsf{div}}(p)$. $w \not\models \phi$. By Proposition 4.7 we also know $w \notin \mathcal{L}_{\mathsf{max}}(B^3_\phi)$, $w \notin \mathcal{L}_{\mathsf{B}}(B^3_\phi)$, and $w \notin \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. We distinguish the following cases.

1. *Case* $w \in \mathcal{L}_{max}(p)$*:* Then, $w \in \mathcal{L}_{\mathsf{max}}(p) \setminus \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. However, $w \notin \mathcal{L}_{\mathsf{max}}(B^3_\phi)$, which contradicts Inclusion (iii).
2. *Case* $w \in \mathcal{L}_B(p)$*:* Hence, $w \in \mathcal{L}_{\mathsf{B}}(p) \setminus \mathcal{L}_{\mathsf{div}}(B^3_\phi)$. However, $w \notin \mathcal{L}_{\mathsf{B}}(B^3_\phi)$, which is a contradiction to Inclusion (iv).
3. *Case* $w \in \mathcal{L}_{div}(p)$*:* But $w \notin \mathcal{L}_{\mathsf{div}}(B^3_\phi)$, which contradicts Inclusion (i).

Thus, direction "$\Longleftarrow$" holds, as desired.